PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

"Structured Assembly Language Programming" <and> Cook

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **Structured Assembly Language Programming** and **Cook**

Found **5,303** of **140,980**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Structured Assembly language in VAX-11 MACRO
Robert R. Leeper, Karl O. Rehmer
February 1986 **ACM SIGCSE Bulletin , Proceedings of the seventeenth SIGCSE technical symposium on Computer science education**, Volume 18 Issue 1
Full text available: pdf(499.69 KB) Additional Information: full citation, abstract, references, index terms

For several years, the introductory assembly language course at Indiana-Purdue at Fort Wayne has used "structured" assembly language on an IBM System 370. A later course makes use of VAX-11 MACRO assembly language on a VAX 11/780. This paper shows how the major constructs for structured programming may be implemented in VAX-11 MACRO. The scheme involves assembly language templates for each of the constructs, a standard labeling scheme, and a commenting method which reflects the ...

2 Advanced problem solving and algorithm development
Robert N. Cook
February 1983 **ACM SIGCSE Bulletin , Proceedings of the fourteenth SIGCSE technical symposium on Computer science education**, Volume 15 Issue 1
Full text available: pdf(273.02 KB) Additional Information: full citation, abstract, references, index terms

A course in advanced problem solving and algorithm development is described in this paper. This course differs from the approach taken in many algorithm texts in that it emphasizes the problem solving process involved in developing the algorithms rather than proofs of correctness. Algorithms are studied in the areas of number theory, array processing, sorting and searching, text processing, and data structures. The course concludes with a discussion of linear, binary, and non-linear recursi ...

3 The suitability of the VAX for a course in assembly language
Robert W. Sebesta
February 1983 **ACM SIGCSE Bulletin , Proceedings of the fourteenth SIGCSE technical symposium on Computer science education**, Volume 15 Issue 1
Full text available: pdf(423.55 KB) Additional Information: full citation, abstract, references, index terms

This paper describes the assembly language course we teach, using a Digital Equipment Corporation VAX-11/780 minicomputer, in which structured programming is stressed. It also discusses the relative merits and disadvantages of choosing the VAX as the computer to be used in such a course. The first section of the paper provides a quick survey of the VAX architecture. The second describes our course in assembly language, including our method of structuring assembly language program ...

4 Schemata for teaching structured assembly language programming
James L. Silver, Robert R. Leeper

February 1983 **ACM SIGCSE Bulletin , Proceedings of the fourteenth SIGCSE technical symposium on Computer science education**, Volume 15 Issue 1
Full text available: pdf(352.19 KB) Additional Information: full citation, abstract, references, index terms

The paper advocates the use of structured programming techniques in designing and developing assembly language programs. It presents schemata for implementing the major constructs of structured programs in IBM 370 Assembly language. These include the extensive use of equates for defining labels and the use of indentation to illustrate logical dependencies in pseudocode comments.

5 Using assembly language to teach concepts in the introductory course
Barry Donahue
February 1988 **ACM SIGCSE Bulletin , Proceedings of the nineteenth SIGCSE technical symposium on Computer science education**, Volume 20 Issue 1
Full text available: pdf(608.81 KB) Additional Information: full citation, abstract, references, citings, index terms

While current trends in the teaching of the introductory course are very positive, several problems still remain to be faced. Among these are: A basic understanding of the operation of a computer. An emphasis on concept development rather than skill development. A proper historical perspective of computer science. To help remove these deficiencies, a very simple virtual machine is introduced. B ...

6 A language for shading and lighting calculations
Pat Hanrahan, Jim Lawson
September 1990 **ACM SIGGRAPH Computer Graphics , Proceedings of the 17th annual conference on Computer graphics and interactive techniques**, Volume 24 Issue 4
Full text available: pdf(2.08 MB) Additional Information: full citation, abstract, references, citings, index terms

A shading language provides a means to extend the shading and lighting formulae used by a rendering system. This paper discusses the design of a new shading language based on previous work of Cook and Perlin. This language has various types of shaders for light sources and surface reflectances, point and color data types, control flow constructs that support the casting of outgoing and the integration of incident light, a clearly specified interface to the rendering system using global state var ...

7 Automatically extracting and representing collocations for language generation
Frank A. Smadja, Kathleen R. McKeown
June 1990 **Proceedings of the 28th conference on Association for Computational Linguistics**
Full text available: pdf(408.21 KB) Publisher Site Additional Information: full citation, abstract, references, citings

Collocational knowledge is necessary for language generation. The problem is that collocations come in a large variety of forms. They can involve two, three or more words, these words can be of different syntactic categories and they can be involved in more or less rigid ways. This leads to two main difficulties: collocational knowledge has to be acquired and it must be represented flexibly so that it can be used for language generation. We address both problems in this paper, focusing on the ac ...

8 Toward a typed foundation for method specialization and inheritance
John C. Mitchell
December 1989 **Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Full text available: pdf(1.69 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper discusses the phenomenon of method specialization in object-oriented programming languages. A typed function calculus of objects and classes is presented, featuring method specialization when methods are added or redefined. The soundness of the typing rules (without subtyping) is suggested by a translation into a more traditional calculus with recursively-defined record types. However, semantic questions regarding the subtype relation on classes remain open.

9 Workshop and conference summaries: Summary: ICSE workshop on dynamic analysis (WODA 2003)
Jonathan E. Cook, Michael D. Ernst
November 2003 **ACM SIGSOFT Software Engineering Notes**, Volume 28 Issue 6
Full text available: pdf(38.27 KB) Additional Information: full citation, abstract, references

Dynamic analysis of software systems has long proven to be a practical approach to gain understanding of the operational behavior of the system. This workshop brought together researchers in the field of dynamic analysis to discuss the breadth of the field, order the field along logical dimensions, expose common issues and approaches, and stimulate synergistic collaborations among the participants.

10 Applications: A Wavelet Packet representation of audio signals for music genre classification using different ensemble and feature selection techniques
Marco Grimaldi, Pádraig Cunningham, Anil Kokaram
November 2003 **Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval**
Full text available: pdf(363.76 KB) Additional Information: full citation, abstract, references, index terms

The vast amount of music available electronically presents considerable challenges for information retrieval. There is a need to annotate music items with descriptors in order to facilitate retrieval. In this paper we present a process for determining the music genre of an item using a new set of descriptors. A Wavelet Packet Transform is applied to obtain the signal representation at different levels. Time and frequency features are extracted from these levels taking into account the nature of ...

Keywords: Wavelet analysis, ensemble techniques, features selection, music information retrieval

11 Particle transport and image synthesis
James Arvo, David Kirk
September 1990 **ACM SIGGRAPH Computer Graphics , Proceedings of the 17th annual conference on Computer graphics and interactive techniques**, Volume 24 Issue 4
Full text available: pdf(470.13 KB) Additional Information: full citation, abstract, references, citings, index terms

The rendering equation is similar to the linear Boltzmann equation which has been widely studied in physics and nuclear engineering. Consequently, many of the powerful techniques which have been developed in these fields can be applied to problems in image synthesis. In this paper we adapt several statistical techniques commonly used in neutron transport to stochastic ray tracing and, more generally, to Monte Carlo solution of the rendering equation. First, we describe a technique known as *Ru* ...

12 Type substitution for object-oriented programming
Jens Palsberg, Michael I. Schwartzbach
September 1990 **ACM SIGPLAN Notices , Proceedings of the European conference on object-oriented programming on Object-oriented programming systems, languages, and applications**, Volume 25 Issue 10
Full text available: pdf(989.92 KB) Additional Information: full citation, abstract, references, citings, index terms

Genericity allows the substitution of types in a class. This is usually obtained through parameterized classes, although they are inflexible since any class can be inherited but is not in itself parameterized. We suggest a new genericity mechanism, type substitution, which is a subclassing concept that complements inheritance: any class is generic, can be "instantiated" gradually without planning, and has all of its generic instances as subclasses.

13 Some applications of logic to feasibility in higher types
Aleksandar Ignjatovic, Arun Sharma
April 2004 **ACM Transactions on Computational Logic (TOCL)**, Volume 5 Issue 2
Full text available: pdf(137.82 KB) Additional Information: full citation, abstract, references, index terms

While it is commonly accepted that computability on a Turing machine in polynomial time represents a correct formalization of the notion of a *feasibly computable* function, there is no similar agreement on how to extend this notion on *functionals*, that is, what functionals should be considered feasible. One possible paradigm was introduced by Mehlhorn, who extended Cobham's definition of feasible functions to type 2 functionals. Subsequently, this class of functionals (with inessent ...

Keywords: bounded arithmetic, functionals, higher-order complexity, second-order theories

14 Analytic response time model for distributed systems
Janice H. Cook, Leo H. Groner
May 1990 **ACM SIGAPL APL Quote Quad , Conference proceedings on APL 90: for the future**, Volume 20 Issue 4
Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, index terms

Network designers are faced with a combinatorial explosion of choices not only among various vendors' workstations, hosts and servers, but also among application distribution strategies, communication media, communication protocols, and network topologies. To study performance trade-offs among various designs, the authors have developed a generic system thruput and response time model for distributed systems. We have applied the model to actual customer networks. The modeller des ...

15 Groupware: some issues and experiences
Clarence A. Ellis, Simon J. Gibbs, Gail Rein
January 1991 **Communications of the ACM**, Volume 34 Issue 1
Full text available: pdf(7.22 MB) Additional Information: full citation, references, citings, index terms

16 An extension of standard ML modules with subtyping and inheritance
John Mitchell, Sigurd Meldal, Neel Madhav
January 1991 **Proceedings of the 18th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Full text available: pdf(1.10 MB) Additional Information: full citation, references, citings, index terms

17 Collaborative technology and group process feedback: their impact on interactive sequences in meetings
Marcial Losada, Pedro Sanchez, Elizabeth E. Noble
September 1990 **Proceedings of the 1990 ACM conference on Computer-supported cooperative work**
Full text available: pdf(1.11 MB) Additional Information: full citation, abstract, references, citings, index terms

We analyzed group collaborative behavior by detecting patterns of interactive sequences in meetings using time series analysis. This is in contrast to previous work in which frequency counts of interactions were analyzed. Researchers have reported a decrease of these interaction frequencies associated with the use of computer-supported collaborative technology [Appl86, McGu87, Sieg86, Wats88]. We found that if group process feedback is given to people participating in a computer-supported c ...

18 Computational complexity and grammatical formalisms: Context-freeness and the computer processing of human languages
Geoffrey K. Pullum
June 1983 **Proceedings of the 21st conference on Association for Computational Linguistics**
Full text available: pdf(628.62 KB) Publisher Site
Additional Information: full citation, abstract, references, citings

Context-free grammars, far from having insufficient expressive power for the description of human languages, may be overly powerful, along three dimensions; (i) weak generative capacity: there exists an interesting proper subset of the CFL's, the profligate CFL's, within which no human language appears to fall; (2) strong generative capacity: human languages can be appropriately described in terms of a proper subset of the CF-PSG's, namely those with the ECPO property; (3) time complexity: the r ...

19 Inheritance is not subtyping
William R. Cook, Walter Hill, Peter S. Canning
December 1989 **Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Full text available: pdf(1.24 MB)
Additional Information: full citation, abstract, references, citings, index terms

In typed object-oriented languages the subtype relation is typically based on the inheritance hierarchy. This approach, however, leads either to insecure type-systems or to restrictions on inheritance that make it less flexible than untyped Smalltalk inheritance. We present a new typed model of inheritance that allows more of the flexibility of Smalltalk inheritance within a statically-typed system. Significant features of our analysis are the introduction of polymorphism into the typing of ...

20 Traits: Tools and Methodology
May 2004 **Proceedings of the 26th International Conference on Software Engineering**
Full text available: pdf(539.73 KB) Publisher Site
Additional Information: full citation, abstract

Traits are an object-oriented programming language constructthat allow groups of methods to be named and reusedin arbitrary places in an inheritance hierarchy. Classes canuse methods from traits as well as defining their own methodsand instance variables. Traits thus enable a new styleof programming, in which traits rather than classes are theprimary unit of reuse. However, the additional sub-structureprovided by traits is always optional: a class written usingtraits can also be viewed as a flat ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Structured programming in assembly language

Full text Pdf (871 KB)

Source **ACM SIGCSE Bulletin** archive
Volume 8 , Issue 4 (December 1976) table of contents
Pages: 53 - 67
Year of Publication: 1976
ISSN:0097-8418

Authors Andries van Dam
Jens M. Dill
Douglas F. Dixon
David S. Notkin

Publisher ACM Press New York, NY, USA

DOI Bookmark: Use this link to bookmark this Article: http://doi.acm.org/10.1145/382222.382464
What is a DOI?

ABSTRACT

Structured design and programming techniques can be extended from high-level languages to assembly language. Over the past three years at Brown University, beginning assembly language programmers have been successfully taught these techniques using clearly defined standards. These standards and the solutions to several of the typical problems that arise in structured assembly language programming are discussed in this paper.

CITINGS *3*

Robert R. Leeper , Karl O. Rehmer, Structured Assembly language in VAX-11 MACRO, Papers of the seventeenth SIGCSE technical symposium on Computer science education, p.53-60, February 06-07, 1986, Cincinnati, Ohio, United States

Karen Mackey , Howard Fosdick, An applied computer science/systems programming approach to teaching data structures, The papers of the 10th SIGCSE technical symposium on Computer science education, p.76-78, January 1979

Barry Donahue, Using assembly language to teach concepts in the introductory course, ACM SIGCSE Bulletin, v.20 n.1, p.158-162, Feb. 1988

INDEX TERMS

Primary Classification:
D. Software

↪ **D.3** PROGRAMMING LANGUAGES
↪ **D.3.2** Language Classifications
↪ **Subjects:** Macro and assembly languages

Additional Classification:

D. Software
↪ **D.2** SOFTWARE ENGINEERING
↪ **D.2.3** Coding Tools and Techniques
↪ **Subjects:** Structured programming
↪ **D.3** PROGRAMMING LANGUAGES

K. Computing Milieux
↪ **K.3** COMPUTERS AND EDUCATION
↪ **K.3.2** Computer and Information Science Education
↪ **Subjects:** Curriculum

General Terms:
Design, Human Factors, Languages, Management, Performance, Theory

Collaborative Colleagues:

Jens M. Dill:	Douglas F. Dixon, David S. Notkin, Andries van Dam
Douglas F. Dixon:	Jens M. Dill, David S. Notkin, Andries van Dam
David S. Notkin:	Jens M. Dill, Douglas F. Dixon, Andries van Dam
Andries van Dam:	Daniel G. Aliaga, David Arnold, Carl Bass, Richard J. Beach, Dan Bergeron, Wes Bethel, Kellogg Booth, Jack Bresenham, Ken Brodlie, Frederick P. Brooks, Steve Bryson, Rick Carey, George S. Carson, Steve Carson, Sharon Rose Clay, Brookshire D. Conner, D. B Conner, D. Brookshire Conner, Steven J. DeRose, Jens M. Dill, Douglas F. Dixon, Louis Doctor, Jonathan L. Elion, Thomas Faulhaber, Steven K. Feiner, Tony Fields, James D. Foley, Andrew Forsberg, Andrew S. Forsberg, Julian E. Gómez, Nahum Gershon, Michael Gleicher, Julian E. Gomez, Daniel L. Gould, Robert Gurwitz, Jan Hardenbergh, Michael Hawley, Lofton R. Henderson, Kenneth Herndon, Kenneth P. Herndon, Brian Hook, Nathan T. Huang, Philip Hubbard, Philip M. Hubbard, Carol Hunter, Michael T. Jones, David Kamins, George Karniadakis, George E. Karniadakis, Arie Kaufman, Henry Kaufman, Mike Kirby, Robert M. Kirby, Brian Knep, David Laidlaw, David H. Laidlaw, Michael Lesk, Dan Mapes, Aaron Marcus, Norman Meyerowitz, Norman Meyrowitz, Janet Michel, Elli Mylonas, William M. Newman, D A Niguidula, David S. Notkin, Randy Pausch, Dick Phillips, Dick Puk, Allen Renear, Bill Ribarsky, David E. Rice, Daniel C. Robbins, Warren Robinett, Larry Rosenblum, David Schlegel, Rosemary Simpson, Rosemary M. Simpson, Scott S. Snibbe, Henry Sowizral, John Tartar, Craig Upson, Dan Venolia, Jefrey Vroom, Matthias M. Wloka, Nicole Yankelovich, Robert C. Zeleznik, Paul van Binst

Jonathan Elion John F. Hughes

Structured Assembly language in VAX-11 MACRO

Full text Pdf (500 KB)

Source **Technical Symposium on Computer Science Education** archive
Proceedings of the seventeenth SIGCSE technical symposium on Computer science education table of contents
Cincinnati, Ohio, United States
Pages: 53 - 60
Year of Publication: 1986
ISSN:0097-8418
Also published in ...

Authors Robert R. Leeper Purdue Univ. at Fort Wayne, IN
Karl O. Rehmer Purdue Univ. at Fort Wayne, IN

Sponsor SIGCSE: ACM Special Interest Group on Computer Science Education
Publisher ACM Press New York, NY, USA

Additional Information: abstract references index terms collaborative colleagues

Tools and Actions: Discussions Find similar Articles Review this Article
Save this Article to a Binder Display in BibTex Format

DOI Bookmark: Use this link to bookmark this Article: http://doi.acm.org/10.1145/5600.5632
What is a DOI?

ABSTRACT

For several years, the introductory assembly language course at Indiana-Purdue at Fort Wayne has used "structured" assembly language on an IBM System 370. A later course makes use of VAX-11 MACRO assembly language on a VAX 11/780. This paper shows how the major constructs for structured programming may be implemented in VAX-11 MACRO. The scheme involves assembly language templates for each of the constructs, a standard labeling scheme, and a commenting method which reflects the structure of the program.

REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. ACM has opted to expose the complete List rather than only correct and linked references.

1 Sara Baase, VAX assembly language (2nd ed.), Prentice-Hall, Inc., Upper Saddle River, NJ, 1992

2 Cook, Robert N., "Structured Assembly Language Programming", SIGCSE Bulletin 14, 1 (February 1982), Pages 193-200.

3 Silver, James L. Jr. and Leeper, Robert R., "Constructs for Structured Assembly Language Programming', SIGCSE Bulletin 15, 1 (February 1983), Pages 128-132.

4 Andries van Dam , Jens M. Dill , Douglas F. Dixon , David S. Notkin, Structured programming in assembly language, ACM SIGCSE Bulletin, v.8 n.4, p.53-67, Dec. 1976

INDEX TERMS

Classification:
D. Software
D.3 PROGRAMMING LANGUAGES
D.3.2 Language Classifications
Subjects: Macro and assembly languages

General Terms:
Languages

Collaborative Colleagues:

Robert R. Leeper:	Robert A. Barret
	Karl O. Rehmer
	James L. Silver
	Spotswood D. Stoddard
Karl O. Rehmer:	Robert R. Leeper

This Article has also been published in:

- **ACM SIGCSE Bulletin**
  Volume 18 , Issue 1 (February 1986)

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

condition code <and> "Structured Assembly language" <and>

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **condition code** and **Structured Assembly language**

Found **77,841** of **140,980**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Structured Assembly language in VAX-11 MACRO
Robert R. Leeper, Karl O. Rehmer
February 1986 **ACM SIGCSE Bulletin , Proceedings of the seventeenth SIGCSE technical symposium on Computer science education**, Volume 18 Issue 1
Full text available: pdf(499.69 KB) Additional Information: full citation, abstract, references, index terms

For several years, the introductory assembly language course at Indiana-Purdue at Fort Wayne has used "structured" assembly language on an IBM System 370. A later course makes use of VAX-11 MACRO assembly language on a VAX 11/780. This paper shows how the major constructs for structured programming may be implemented in VAX-11 MACRO. The scheme involves assembly language templates for each of the constructs, a standard labeling scheme, and a commenting method which reflects the ...

2 Avoiding conditional branches by code replication
Frank Mueller, David B. Whalley
June 1995 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1995 conference on Programming language design and implementation**, Volume 30 Issue 6
Full text available: pdf(1.10 MB) Additional Information: full citation, references, citings, index terms

3 The PDP-11: A case study of how not to design condition codes
Robert D. Russell
April 1978 **Proceedings of the 5th annual symposium on Computer architecture**
Full text available: pdf(503.09 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper investigates a design weakness in the PDP-11 architecture, namely the condition code bits. Experience with the machine has demonstrated a number of "traps" for the unwary programmer stemming directly from an inconsistent and sometimes confusing scheme of condition code settings. This is particularly annoying in view of the otherwise clean architectural characteristics of the machine. A number of "principles" are proposed that would correct the deficiencies ...

4 Selection conditions in main memory
Kenneth A. Ross
March 2004 **ACM Transactions on Database Systems (TODS)**, Volume 29 Issue 1
Full text available: pdf(296.54 KB) Additional Information: full citation, abstract, references, index terms

We consider the fundamental operation of applying a compound filtering condition to a set of records. With large main memories available cheaply, systems may choose to keep the data entirely in main memory, in order to improve query and/or update performance.The design of a data-intensive algorithm in main memory needs to take into account the architectural characteristics of modern processors, just as a disk-based method needs to consider the physical characteristics of disk devices. An importa ...

Keywords: Branch misprediction

5 FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation
Nicholas J. Higham
December 1988 **ACM Transactions on Mathematical Software (TOMS)**, Volume 14 Issue 4
Full text available: pdf(1.10 MB) Additional Information: full citation, abstract, references, citings, index terms

FORTRAN 77 codes SONEST and CONEST are presented for estimating the 1-norm (or the infinity-norm) of a real or complex matrix, respectively. The codes are of wide applicability in condition estimation since explicit access to the matrix, A, is not required; instead, matrix-vector products Ax and ATx are computed by the calling program via a reverse communication interface. The algorithms are based on a convex optimizat ...

Keywords: FORTRAN, LINPACK, condition estimation, condition number, convex optimization, infinity-norm, one-norm, reverse communication

6 Testing and coverage: Elided conditionals
Manos Renieris, Sébastien Chan-Tin, Steven P. Reiss
June 2004 **Proceedings of the ACM-SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering**
Full text available: pdf(95.18 KB) Additional Information: full citation, abstract, references, index terms

Many software testing and automated debugging tools rely on structural coverage techniques. Such tools implicitly assume a relation between individual control-flow choices made in conditional statements during a program run and the outcome of the run. In this paper, we develop the notion of *elided choices* that, viewed in isolation, have no impact on the outcome of the program. We call the conditionals that make such choices *elided conditionals*. We develop an experimental framework ...

Keywords: coverage testing, dynamic mutation, elision

7 Should SCC set condition codes?
Fleur Liane Williams
September 1988 **ACM SIGARCH Computer Architecture News**, Volume 16 Issue 4
Full text available: pdf(247.58 KB) Additional Information: full citation, index terms

8 Automated Generation of Test Programs from Closed Specifications of Classes and Test Cases
May 2004 **Proceedings of the 26th International Conference on Software Engineering**
Full text available: pdf(315.07 KB) Publisher Site Additional Information: full citation, abstract

Most research on automated specification-based softwaretesting has focused on the automated generation oftest cases. Before a software system can be tested, it must beset up according to the input requirements of the test cases.This setup process is usually performed manually, especiallywhen testing complex data structures and databases.After the system is properly set up, a test execution tool runsthe system according to the test cases and pre-recorded testscripts to obtain the outputs, which a ...

9 A foundation for sequentializing parallel code
B. Simons, D. Alpern, J. Ferrante
May 1990 **Proceedings of the second annual ACM symposium on Parallel algorithms and architectures**
Full text available: pdf(1.16 MB) Additional Information: full citation, references, citings, index terms

10 A geometric approach to integer condition codes and branch instructions
James W. Benham
December 1995 **ACM SIGCSE Bulletin**, Volume 27 Issue 4
Full text available: pdf(437.86 KB) Additional Information: full citation, index terms

11 Security and privacy: Securing web application code by static analysis and runtime protection
Yao-Wen Huang, Fang Yu, Christian Hang, Chung-Hung Tsai, Der-Tsai Lee, Sy-Yen Kuo
May 2004 **Proceedings of the 13th conference on World Wide Web**
Full text available: pdf(608.77 KB) Additional Information: full citation, abstract, references, index terms

Security remains a major roadblock to universal acceptance of the Web for many kinds of transactions, especially since the recent sharp increase in remotely exploitable vulnerabilities have been attributed to Web application bugs. Many verification tools are discovering previously unknown vulnerabilities in legacy C programs, raising hopes that the same success can be achieved with Web applications. In this paper, we describe a sound and holistic approach to ensuring Web application security. Vi ...

Keywords: information flow, noninterference, program security, security vulnerabilities, type systems, verification, web application security

12 A framework for the integration of partial evaluation and abstract interpretation of logic programs
Michael Leuschel
May 2004 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 26 Issue 3
Full text available: pdf(319.71 KB) Additional Information: full citation, abstract, references, index terms

Recently the relationship between abstract interpretation and program specialization has received a lot of scrutiny, and the need has been identified to extend program specialization techniques so as to make use of more refined abstract domains and operators. This article clarifies this relationship in the context of logic programming, by expressing program specialization in terms of abstract interpretation. Based on this, a novel specialization framework, along with generic correctness results ...

Keywords: Partial deduction, abstract interpretation, flow analysis, logic programming, partial evaluation, program transformation

13 The production of code-mixed discourse
David Sankoff
August 1998
Full text available: pdf(1.40 MB) Publisher Site Additional Information: full citation, abstract, references

We propose a comprehensive theory of code-mixed discourse, encompassing equivalencepoint and insertional code-switching, palindromic constructions and lexical borrowing. The starting point is a production model of code-switching acconting for empirical observations about switch-point distribution (the equivalence constraint), well-formedness of monolingual fragments, conservation of constituent structure and lack of constraint between successive switch points, without invoking any "code-switchin ...

14 Technical contributions: A conditional critical region pre-processor for C based on the Owicki and Gries scheme
Michael F. Kilian
April 1985 **ACM SIGPLAN Notices**, Volume 20 Issue 4
Full text available: pdf(648.25 KB) Additional Information: full citation, references

15 KISS: keep it simple and sequential
Shaz Qadeer, Dinghao Wu
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6
Full text available: pdf(204.52 KB) Additional Information: full citation, abstract, references, index terms

The design of concurrent programs is error-prone due to the interaction between concurrently executing threads. Traditional automated techniques for finding errors in concurrent programs, such as model checking, explore all possible thread interleavings. Since the number of thread interleavings increases exponentially with the number of threads, such analyses have high computational complexity. In this paper, we present a novel analysis technique for concurrent programs that avoids this exponent ...

Keywords: assertion checking, concurrent software, model checking, program analysis, race detection

16 Technical contributions: Syntax extension and the IMP72 programming language
Walter Bilofsky
May 1974 **ACM SIGPLAN Notices**, Volume 9 Issue 5
Full text available: pdf(765.70 KB) Additional Information: full citation, abstract, references

The IMP72 language for the DEC PDP-10 computer is the most recent of the IMP family of extensible software implementation languages. Its facility for extending the syntax of the language, the syntax statement, is simple enough to be useful to relatively unsophisticated users, yet powerful enough that the IMP72 language is defined by an IMP72 source program consisting of syntax statements.

Keywords: extensible language, implementation language, syntax extension, syntax-directed compiler

17 Morphing aspects: incompletely woven aspects and continuous weaving
Stefan Hanenberg, Robert Hirschfeld, Rainer Unland
March 2004 **Proceedings of the 3rd international conference on Aspect-oriented software development**
Full text available: pdf(3.73 MB) Additional Information: full citation, abstract, references, citings

Weaving is one of the fundamental mechanisms of aspect-oriented systems. A weaver composes different aspects with the base system by determining and adapting all parts where aspect specific elements are needed eventually. At runtime, timeconsuming join point checks are necessary to determine if at a certain join point aspect-specific code needs to be executed. Current technologies enforce such checks even in locations that only temporarily or under restrictive conditions (or even never) execute ...

18 Building adaptive distributed applications with middleware and aspects
Gary Duzan, Joseph Loyall, Richard Schantz, Richard Shapiro, John Zinky
March 2004 **Proceedings of the 3rd international conference on Aspect-oriented software development**

Full text available: pdf(1.07 MB) Additional Information: full citation, abstract, references, index terms

Middleware technologies allow the development of distributed applications without explicit knowledge of the networking involved. However, in the face of changing network and CPU conditions across the distributed system, these applications often will need to adapt their behavior to maintain an acceptable quality of service (QoS), which implies a knowledge of these conditions. This adaptation is neither part of the application logic nor part of the distribution middleware, and so represents a sepa ...

Keywords: CORBA, adaptive quality of service, aspect-oriented programming, distributed objects, middleware

19 Profiling Java applications using code hotswapping and dynamic call graph revelation
Mikhail Dmitriev
January 2004 **ACM SIGSOFT Software Engineering Notes , Proceedings of the fourth international workshop on Software and performance**, Volume 29 Issue 1
Full text available: pdf(1.32 MB) Additional Information: full citation, abstract, references

Instrumentation-based profiling has many advantages and one serious disadvantage: usually high performance overhead. This overhead can be substantially reduced if only a small part of the target application (for example, one that has previously been identified as a performance bottleneck) is instrumented, while the rest of the application code continues to run at full speed. The value of such a profiling technology would increase further if the code could be instrumented and de-instrumented as m ...

20 Informatics: input and output: An assignment of key-codes for a Japanese character keyboard
Yuzuru Hiraga, Yoshihiko Ono, Yamada-Hisao
September 1980 **Proceedings of the 8th conference on Computational linguistics**
Full text available: pdf(798.47 KB) Additional Information: full citation, references

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

PORTAL Subscribe Register Login
(Full Service) (Limited Service, Free)

Search: ● The ACM Digital Library ○ The Guide

THE ACM DIGITAL LIBRARY Feedback Report a probler

Schemata for teaching structured assembly language programming

Full text Pdf (352 KB)

Source **Technical Symposium on Computer Science Education** archive

Proceedings of the fourteenth SIGCSE technical symposium on Computer science edu

Orlando, Florida, United States
Pages: 128 - 132
Year of Publication: 1983
ISBN:0-89791-091-5
Also published in ...

Authors James L. Silver, Jr.
Robert R. Leeper

Sponsors SIGCSE: ACM Special Interest Group on Computer Science Education
ACM: Association for Computing Machinery

Publisher ACM Press New York, NY, USA

Additional Information: abstract references index terms collaborative colleagues

Tools and Actions: Discussions
Find similar Articles
Review this Article

Save this Article to a Binder
Display in BibTex Format

DOI Bookmark: Use this link to bookmark this Article: http://doi.acm.org/10.1145/800038.801
What is a DOI?

ABSTRACT

The paper advocates the use of structured programming techniques in designing and develc
programs. It presents schemata for implementing the major constructs of structured progra
language. These include the extensive use of equates for defining labels and the use of inde
dependencies in pseudocode comments.

↑ REFERENCES

Note: OCR errors may be found in this Reference List extracted from the full text article. AC
complete List rather than only correct and linked references.

1 Richard H. Austing , Bruce H. Barnes , Della T. Bonnette , Gerald L. Engel , Gordon Stok
recommendations for the undergraduate program in computer science&mdash; a report of t
committee on computer science, Communications of the ACM, v.22 n.3, p.147-166, March 1

2 Per Brinch Hansen, Operating System Principles, Prentice Hall PTR, Upper Saddle River,

3 Cook, Robert N., "Structured Assembly Language Programming", SIGCSE Bulletin 14, 1(
193-200.

↑ INDEX TERMS

Primary Classification:
D. Software
↪ D.3 PROGRAMMING LANGUAGES
↪ D.3.2 Language Classifications
↪ Subjects: Macro and assembly languages

Additional Classification:
D. Software
↪ D.2 SOFTWARE ENGINEERING
↪ D.2.3 Coding Tools and Techniques
↪ Subjects: Structured programming

K. Computing Milieux
↪ K.3 COMPUTERS AND EDUCATION
↪ K.3.2 Computer and Information Science Education
↪ Subjects: Curriculum

General Terms:
Languages

↑ Collaborative Colleagues:

Robert R. Leeper:	Robert A. Barret
	Karl O. Rehmer
	James L. Silver
	Spotswood D. Stoddard
James L. Silver:	Robert R. Leeper

↑ This Article has also been published in:

- **ACM SIGCSE Bulletin**
  Volume 15 , Issue 1 (February 1983)

PORTAL Subscribe Register Login
(Full Service) (Limited Service, Free)

Search: ● The ACM Digital Library ○ The Guide
"Structured Assembly Language" <and> condition code

THE ACM DIGITAL LIBRARY Feedback

Terms used Structured Assembly Language and condition code

Sort results by relevance

Display results expanded form

Save results to a Binder Try
Search Tips Try
☐ Open results in a new window

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9
Best 200 shown

1 Structured Assembly language in VAX-11 MACRO
Robert R. Leeper, Karl O. Rehmer
February 1986 ACM SIGCSE Bulletin , Proceedings of the seventeenth SIGCSE tech education, Volume 18 Issue 1
Full text available: pdf(499.69 KB) Additional Information: full citation, abstract, referen

For several years, the introductory assembly language course at Indiana-Purd &ldquo;structured&rdquo; assembly language on an IBM System 370. A later assembly language on a VAX 11/780. This paper shows how the major constru implemented in VAX-11 MACRO. The scheme involves assembly language tem standard labeling scheme, and a commenting method which reflects the ...

2 Technical contributions: Omitted portions of paper: AL: a structured asseml
Edward C. Haines
April 1973 ACM SIGPLAN Notices, Volume 8 Issue 4
Full text available: pdf(219.37 KB) Additional Information: full citation, references

3 **Schemata for teaching structured assembly language programming**
James L. Silver, Robert R. Leeper
February 1983 ACM SIGCSE Bulletin , Proceedings of the fourteenth SIGCSE techn
education, Volume 15 Issue 1
Full text available: pdf(352.19 KB) Additional Information: full citation, abstract, referen

The paper advocates the use of structured programming techniques in designi
programs. It presents schemata for implementing the major constructs of stru
language. These include the extensive use of equates for defining labels and t
dependencies in pseudocode comments.

4 **The suitability of the VAX for a course in assembly language**
Robert W. Sebesta
February 1983 ACM SIGCSE Bulletin , Proceedings of the fourteenth SIGCSE techn
education, Volume 15 Issue 1
Full text available: pdf(423.55 KB) Additional Information: full citation, abstract, referen

This paper describes the assembly language course we teach, using a Digital I
minicomputer, in which structured programming is stressed. It also discusses
choosing the VAX as the computer to be used in such a course. The first sectio
of the VAX architecture. The second describes our course in assembly languag
assembly language program ...

5 **Using assembly language to teach concepts in the introductory course**
Barry Donahue
February 1988 ACM SIGCSE Bulletin , Proceedings of the nineteenth SIGCSE techn
education, Volume 20 Issue 1
Full text available: pdf(608.81 KB) Additional Information: full citation, abstract, references,

While current trends in the teaching of the introductory course are very positi
faced. Among these are: A basic understanding of the operation of a compute
rather than skill development. A proper historical perspective of computer scie
a very simple virtual machine is introduced. B ...

6 **Avoiding conditional branches by code replication**
Frank Mueller, David B. Whalley
June 1995 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1995 confere
and implementation, Volume 30 Issue 6
Full text available: pdf(1.10 MB) Additional Information: full citation, references, citings

7 The PDP-11: A case study of how not to design condition codes
Robert D. Russell
April 1978 Proceedings of the 5th annual symposium on Computer architectu
Full text available: pdf(503.09 KB) Additional Information: full citation, abstract, references,

This paper investigates a design weakness in the PDP-11 architecture, namely
with the machine has demonstrated a number of &ldquo;traps&rdquo; for the
from an inconsistent and sometimes confusing scheme of condition code setti
of the otherwise clean architectural characteristics of the machine. A number
proposed that would correct the deficiencies ...

8 Selection conditions in main memory
Kenneth A. Ross
March 2004 ACM Transactions on Database Systems (TODS), Volume 29 Iss
Full text available: pdf(296.54 KB) Additional Information: full citation, abstract, referen

We consider the fundamental operation of applying a compound filtering cond
memories available cheaply, systems may choose to keep the data entirely in
query and/or update performance.The design of a data-intensive algorithm in
account the architectural characteristics of modern processors, just as a disk-l
physical characteristics of disk devices. An importa ...

Keywords: Branch misprediction

9 FORTRAN codes for estimating the one-norm of a real or complex matrix,
estimation
Nicholas J. Higham
December 1988 ACM Transactions on Mathematical Software (TOMS), Volum
Full text available: pdf(1.10 MB) Additional Information: full citation, abstract, references, cit

FORTRAN 77 codes SONEST and CONEST are presented for estimating the 1-n
complex matrix, respectively. The codes are of wide applicability in condition
matrix, A, is not required; instead, matrix-vector products Ax and ATx are con
reverse communication interface. The algorithms are based on a convex optin

Keywords: FORTRAN, LINPACK, condition estimation, condition number, conve
one-norm, reverse communication

10 Testing and coverage: Elided conditionals
Manos Renieris, Sébastien Chan-Tin, Steven P. Reiss
June 2004 Proceedings of the ACM-SIGPLAN-SIGSOFT workshop on Program anal
Full text available: pdf(95.18 KB) Additional Information: full citation, abstract, reference

Many software testing and automated debugging tools rely on structural cove
assume a relation between individual control-flow choices made in conditional
the outcome of the run. In this paper, we develop the notion of *elided choices*
impact on the outcome of the program. We call the conditionals that make su
develop an experimental framework ...

Keywords: coverage testing, dynamic mutation, elision

11 Should SCC set condition codes?
Fleur Liane Williams
September 1988 ACM SIGARCH Computer Architecture News, Volume 16 Issue 4
Full text available: pdf(247.58 KB) Additional Information: full citation, index terms

12 Automated Generation of Test Programs from Closed Specifications of Cla
May 2004 Proceedings of the 26th International Conference on Software Engir
Full text available: pdf(315.07 KB) Publisher Site Additional Informatic

Most research on automated specification-based softwaretesting has focused c
cases. Before a software system can be tested, it must beset up according to t
cases.This setup process is usually performed manually, especiallywhen testir
databases.After the system is properly set up, a test execution tool runsthe sy
pre-recorded testscripts to obtain the outputs, which a ...

13 A foundation for sequentializing parallel code
B. Simons, D. Alpern, J. Ferrante
May 1990 Proceedings of the second annual ACM symposium on Parallel algorithn
Full text available: pdf(1.16 MB) Additional Information: full citation, references, citings, index ter

14 A geometric approach to integer condition codes and branch instructions
James W. Benham
December 1995 ACM SIGCSE Bulletin, Volume 27 Issue 4
Full text available: pdf(437.86 KB) Additional Information: full citation, index terms

15 Security and privacy: Securing web application code by static analysis and
Yao-Wen Huang, Fang Yu, Christian Hang, Chung-Hung Tsai, Der-Tsai Lee, Sy-Y
May 2004 Proceedings of the 13th conference on World Wide Web
Full text available: pdf(608.77 KB) Additional Information: full citation, abstract, referenc

Security remains a major roadblock to universal acceptance of the Web for ma
since the recent sharp increase in remotely exploitable vulnerabilities have be
Many verification tools are discovering previously unknown vulnerabilities in l
the same success can be achieved with Web applications. In this paper, we de
ensuring Web application security. Vi ...

Keywords: information flow, noninterference, program security, security vulne
web application security

16 A framework for the integration of partial evaluation and abstract interpretat
Michael Leuschel
May 2004 ACM Transactions on Programming Languages and Systems (TOPLAS
Full text available: pdf(319.71 KB) Additional Information: full citation, abstract, referen

Recently the relationship between abstract interpretation and program special
and the need has been identified to extend program specialization techniques
abstract domains and operators. This article clarifies this relationship in the co
expressing program specialization in terms of abstract interpretation. Based o
along with generic correctness results ...

Keywords: Partial deduction, abstract interpretation, flow analysis, logic progr
transformation

17 The production of code-mixed discourse
David Sankoff
August 1998
Full text available: pdf(1.40 MB) Publisher Site Additional Information: full citation

We propose a comprehensive theory of code-mixed discourse, encompassing e
code-switching, palindromic constructions and lexical borrowing. The starting
code-switching acconting for empirical observations about switch-point distrib
well-formedness of monolingual fragments, conservation of constituent structu
successive switch points, without invoking any "code-switchin ...

18 Technical contributions: A conditional critical region pre-processor for C ba
Michael F. Kilian
April 1985 ACM SIGPLAN Notices, Volume 20 Issue 4
Full text available: pdf(648.25 KB) Additional Information: full citation, re

19 KISS: keep it simple and sequential
Shaz Qadeer, Dinghao Wu
June 2004 ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 confere
and implementation, Volume 39 Issue 6
Full text available: pdf(204.52 KB) Additional Information: full citation, abstract, referen

The design of concurrent programs is error-prone due to the interaction betwe
Traditional automated techniques for finding errors in concurrent programs, su
possible thread interleavings. Since the number of thread interleavings increa
threads, such analyses have high computational complexity. In this paper, we
concurrent programs that avoids this exponent ...

Keywords: assertion checking, concurrent software, model checking, program

20 Technical contributions: Syntax extension and the IMP72 programming lan
Walter Bilofsky
May 1974 ACM SIGPLAN Notices, Volume 9 Issue 5
Full text available: pdf(765.70 KB) Additional Information: full citation, abstract,

The IMP72 language for the DEC PDP-10 computer is the most recent of the II
implementation languages. Its facility for extending the syntax of the languag
enough to be useful to relatively unsophisticated users, yet powerful enough t
IMP72 source program consisting of syntax statements.

Keywords: extensible language, implementation language, syntax extension, :

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7

CiteSeer Find: Structured Assembly Language Progi Documents Citations

Searching for PHRASE **structured assembly language programming robert n cook.**
Restrict to: Header Title Order by: Expected citations Hubs Usage Date Try: Google (CiteSeer) Google (Web) CSB DBLP
No documents match Boolean query. Trying non-Boolean relevance query.
500 documents found. **Order: relevance to query.**

A Unifying Framework for Integer and Finite Domain.. - Bockmayr, Kasper (1997) (Correct) (3 citations)
can **n**ow detect that this inequality fits into the **structure** of the setpack constraint and may infer, e.g.
which combines recent progress in **programming language** design, like constraint logic **programming** [23]
for Integer and Finite Domain Constraint **Programming** Alexander Bockmayr Thomas Kasper
www.mpi-sb.mpg.de/~bockmayr/publications/reports/MPI-I-97-2-008.ps.gz

An Initial Assessment of Aspect-oriented Programming - Robert Walker (1998) (Correct) (14 citations)
different sub-**languages** for expressing the **structure** This work has been submitted to the IEEE for
of software system development. **Programming languages** help software engineers explicitly maintain the
An Initial Assessment of Aspect-oriented **Programming Robert** J. Walker, Elisa L.A. Baniassad and
www.cs.ubc.ca/spider/walker/papers/walker.1998c.ps.gz

A Programming Logic for a Verified Structured Assembly Language - Curzon (1992) (Correct) (1 citation)
A **Programming** Logic for a Verified **Structured Assembly Language** Paul Curzon University of
A **Programming** Logic for a Verified **Structured Assembly Language** Paul Curzon University of Cambridge
Logic for a Verified **Structured Assembly Language** Paul Curzon University of Cambridge Computer
www.cl.cam.ac.uk/Research/HVG/vista/PAPERS/lpar.ps.gz

Multi-Language Programming Environments for High Performance Java.. - Getov (1999) (Correct) (2 citations)
based on conventional JVM The initial **structure** of our **programming** environment including all
Multi-**Language Programming** Environments for High Performance
Multi-**Language Programming** Environments for High Performance Java
www.cs.cf.ac.uk/hpjworkshop/vladimir.ps

Statistical Analysis of an Automated Computer Architecture.. - Waldron Horgan And (Correct)
of program style, where style includes program **structure**, choice of identifier **n**ames, maintainability,
computer architecture with a strong emphasis on **assembly language programming** as part of a Graduate
architecture with a strong emphasis on **assembly language programming** as part of a Graduate Diploma in
mars.cs.unp.ac.za/~jwaldron/tmp/paper.ps

Implementation Is Semantic Interpretation - Rapaport (Correct)
an implementation of a computer program 3. a data **structure** is an implementation of an abstract data type
is its **structure** and behavior as seen by an **assembly-language programmer** .The implementation .
of an algorithm, expressed in a **programming language**. Often, it is said that the program
www.cse.buffalo.edu/pub/WWW/faculty/rapaport/Papers/implementation.ps

From System F to Typed Assembly Language - Morrisett, Walker, Crary, Glew (1998) (Correct) (6 citations)
that do **n**ot often benefit from the additional **structure** supplied by a typing discipline [23, 19, 29,
From System F to Typed **Assembly Language** #Greg Morrisett David Walker Cornell
From System F to Typed **Assembly Language** #Greg Morrisett David Walker Cornell University
www.cs.cmu.edu/~crary/papers/1998/tal/tal-long.ps.gz

A Verified Compiler for a Structured Assembly Language - Curzon (1991) (Correct) (1 citation)
A Verified Compiler for a **Structured Assembly Language** Paul Curzon University of
A Verified Compiler for a **Structured Assembly Language** Paul Curzon University of Cambridge
A Verified Compiler for a **Structured Assembly Language** Paul Curzon University of Cambridge Computer
www.cl.cam.ac.uk/Research/HVG/vista/PAPERS/HUG91.ps.gz

The Verified Compilation of Vista Programs - Curzon (1994) (Correct) (7 citations)
microprocessor considered are a subset of the **structured assembly language** Vista, and the VIPER
considered are a subset of the **structured assembly language** Vista, and the VIPER microprocessor,
using the HOL theorem proving system. The **language** and microprocessor considered are a subset of
www.cl.cam.ac.uk/Research/HVG/vista/PAPERS/procos.ps.gz

Space-saving Optimisations for the Glasgow Haskell Compiler - O'Sullivan (1994) (Correct)

Flattening a tree consists of removing its **structure** by traversing it and converting it to list form
a simple analysis program which operates at the **assembly language** level, similar to that described in
University bos@dcs.glasgow.ac.uk Functional **Languages** Implementation Workshop. Paper 28 Abstract
ftp.dcs.gla.ac.uk/pub/glasgow-fp/authors/Bryan_OSullivan/Space_Savings_for_GHC.ps.gz

Designing Parallel Programs by the Graphical Language GRAPNEL - Eter Kacsuk (1996) (Correct) (14 citations)
are possible but can be applied only in a well **structured** manner. GRAPNEL is a hybrid **language**, where the
Designing Parallel Programs by the Graphical **Language** GRAPNEL 3 P' eter Kacsuk, G' abor D' ozsa,
Abstract We propose a new visual **programming language**, called GRAPNEL (GRAphical Process's
www.kfki.hu/~mszkihp/info/ParComp/papers/EuroMicroPSE-grapnel.ps.Z

Code Composition as an Implementation Language for Compilers - Stichnoth, Gross (1997) (Correct) (7 citations)
in with respect to the rest of the compiler. This **structure** is important for code reuse: the same
may illustrate the concept. A compiler takes **assembly-language** or intermediate-code sequences, which
Code Composition as an Implementation **Language** for Compilers James M. Stichnoth and Thomas
pecan.srv.cs.cmu.edu/afs/cs.cmu.edu/user/stichnot/public/www/dsl97.ps

From System F to Typed Assembly Language (Extended Version) - Morrisett, Walker, Crary.. (1997) (Correct)
that do not often benefit from the additional **structure** supplied by a typing discipline [23, 19, 31,
From System F to Typed **Assembly Language** Extended Version) Greg Morrisett
From System F to Typed **Assembly Language** Extended Version) Greg Morrisett David Walker
cs.cornell.edu/home/jgm/papers/tal-tr.ps

On Implementations and Semantics of a Concurrent Programming.. - Sewell (1997) (Correct) (4 citations)
a user, together with their relationship to the **structured** operational semantics (SOS)This discussion
and Semantics of a Concurrent **Programming Language** Peter Sewell 1 Abstract The concurrency theory
On Implementations and Semantics of a Concurrent **Programming Language** Peter Sewell 1 Abstract The
www.cl.cam.ac.uk/users/pes20/pict9-crc11pt.ps.gz

Merging Interactive, Modular, And Object-Oriented Programming - Tung (Correct)
requires an understanding of the problem and the **structure** of its solution. Modular **programming languages**
speed, program size, and system flexibility using **assembly** code with different table lookup strategies
design, and implementation of the imp **language**, the IMP system, and the IMOOP system. The
ftp.cs.indiana.edu/pub/techreports/TR349.ps.Z

Ellie - A General, Fine-Grained, First Class Object Based Language - Andersen (1991) (Correct)
by allowing definitions of new type and control **structure** abstractions and by having a non-strict
A General, Fine-Grained, First Class Object Based **Language** Birger Andersen Department of Computer Science
1991 1. To appear in Journal of Object-Oriented **Programming**. 2 Abstract This paper presents the
www.econ.cbs.dk/people/birger/Ellie/pub/91gen.ps.gz

A Formal Compiler Specification Method - Levin Bounimova (Correct)
with an interpretation domain consisting of a **structured** pair of source and target derivation (parse)
compiler, where the former is an ITU-T standard **language** popular in telecommunication software world and
still advantageous over informal and straight **programming** and debugging approaches. In the context of
ftp.srdc.metu.edu.tr/pub/fmg/papers/a_formal_compiler_specification_method.ps.gz

Explicit Polymorphism and CPS Conversion - Harper (1993) (Correct) (36 citations)
as such is primarily concerned with the static **structure** and properties of programs. Matters of control
Symposium on Principles of **Programming Languages**, Charleston, SC, January, 1993. This research
Acm Sigplan/sigact Symposium On Principles Of **Programming Languages**, Charleston, Sc, January, 1993. This
www.cs.cmu.edu/People/rwh/papers/cps-fomega/tr.ps

Programming Language Semantics in Foundational Type Theory - Crary (1998) (Correct) (1 citation)
Martin-Lof type theory is closely tied to a **structured** operational semantics and has denotational
Karl Crary, And Neal Glew. From System F To Typed **Assembly Language**. In Twenty-Fifth Acm Sigact-Sigplan
Programming Language Semantics in Foundational Type Theory Karl
foxnet.cs.cmu.edu/afs/cs.cmu.edu/project/fox/mosaic/people/crary/papers/1998/tt-semant/tt-semant-tr.ps.gz

First 20 documents Next 20

Try your query at: Google (CiteSeer) Google (Web) CSB DBLP